Microsoft

Bluetooth Design Review

Designing Hardware and Drivers for the Microsoft® Windows® Family of Operating Systems

Windows Whistler IRPs for Bluetooth

MICROSOFT CONFIDENTIAL DRAFT Version 0.7 — June 5, 2000

Abstract

This paper presents preliminary information about Microsoft® Windows® codenamed Windows "Whistler" IRPs for Bluetooth. Developers should use the information in this document for planning purposes. Developers should plan to write final code on the basis of the released Driver Development Kit (DDK) to be provided on http://www.microsoft.com/ddk/.

Contents

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESSED OR IMPLIED, IN THIS DOCUMENT.

Microsoft Corporation may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give you any license to the patents, trademarks, copyrights, or other intellectual property rights except as expressly provided in any written license agreement from Microsoft Corporation.

Microsoft does not make any representation or warranty regarding specifications in this document or any product or item developed based on these specifications. Microsoft disclaims all expressed and implied warranties, including but not limited to the implied warranties or merchantability, fitness for a particular purpose, and freedom from infringement. Without limiting the generality of the foregoing, Microsoft does not make any warranty of any kind that any item developed based on these specifications, or any portion of a specification, will not infringe any copyright, patent, trade secret, or other intellectual property right of any person or entity in any country. It is your responsibility to seek licenses for such intellectual property rights where appropriate. Microsoft shall not be liable for any damages arising out of or in connection with the use of these specifications, including liability for lost profit, business interruption, or any other damages whatsoever. Some states do not allow the exclusion or limitation of liability or consequential or incidental damages; the above limitation may not apply to you.

Microsoft, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

Introduction

A Bluetooth client driver uses Bluetooth request blocks to communicate with the Bluetooth port driver. A Bluetooth request block is packaged in a BRB structure.

To send a BRB, the driver creates an IRP_MJ_INTERNAL_DEVICE_CONTROL request with the I/O control code IOCTL_INTERNAL_BTH_SUBMIT_BRB and a pointer to an initialized BRB at **Parameters.Others.Argument1** in the I/O stack location of the IRP.

Before a client driver can send a BRB, it must register certain required callback routines and can register additional optional callback routines. To register callback routines, the client driver issues a query-interface IRP (IRP_MJ_PNP with IRP_MN_QUERY_INTERFACE) with a pointer to an initialized BTHPORT_INTERFACE structure that contains pointers to the client driver's callback routines. The port driver calls the appropriate routine after it finishes processing an asynchronous BRB such as an HCI inquiry request.

This paper describes Bluetooth structures used in Bluetooth request blocks. Bluetooth request blocks are defined in Bthddi.h.

Readers should be familiar with Bluetooth as defined in the *Specification of the Bluetooth System*, Volume 1, "Core," and Volume 2, "Profiles," which are available for download from: http://www.bluetooth.com/developer/specification/specification.asp

For general information about IRPs and IOCTLs, see the Windows 2000 Driver Development Kit, available online at http://www.microsoft.com/ddk

BRB

```
typedef struct _BRB {
  union {
    struct _BRB_HEADER BrbHeader;
    struct _BRB_INQUIRY BrbInquiry;
    struct _BRB_CANCEL_INQUIRY BrbInquiryCancel;
    struct _BRB_GET_DEVICE_LIST BrbGetDeviceList;
    struct _BRB_GET_LOCAL_BD_ADDR BrbGetLocalBdAddress;
    struct _BRB_GET_SIGNAL_PARAMETERS BrbGetSignalParameters;
    struct _BRB_HCI_UNKNOWN_CMD BrbHciUnknownCmd;
    struct _BRB_FLUSH BrbFlush;
    struct _BRB_SCO_OPEN BrbScoOpen;
    struct _BRB_SCO_CLOSE BrbScoClose;
    struct _BRB_SCO_READ BrbScoRead;
    struct _BRB_SCO_WRITE BrbScoWrite;
    struct _BRB_L2CA_CONNECT_REQ BrbL2caConnectReq;
    struct _BRB_L2CA_CONNECT_RSP BrbL2caConnectRsp;
    struct _BRB_L2CA_DISCONNECT_REQ BrbL2caDisconnectReq;
    struct _BRB_L2CA_DISCONNECT_RSP BrbL2caDisconnectRsp;
    struct _BRB_L2CA_ACL_TRANSFER BrbL2caAclTransfer;
    struct _BRB_L2CA_CONFIG_REQ BrbL2caConfigReq;
    struct _BRB_L2CA_CONFIG_RSP BrbL2caConfigRsp;
    struct _BRB_SDP_SEARCH_SERVICE_RECORD BrbSearchServiceRecord;
    struct _BRB_SDP_SEARCH_SERVICE_ATTRIBUTE BrbSearchServiceAttribute;
    struct _BRB_SDP_SEARCH_SERVICE_RECORD_ATTRIBUTE BrbSearchServiceRecordAttribute;
  };
} BRB, *PBRB;
```

The BRB structure defines a Bluetooth request block.

Members

BrbHeader
Describes the size in bytes, request type, and completion status for a Bluetooth request. Each Bluetooth command structure contains a _BRB_HEADER structure in which the client driver specifies the size and type of the request. A BRB consisting of a standalone _BRB_HEADER structure is reserved for internal use by the Bluetooth port driver.

BrbInquiry
Triggers the radio to issue an HCI Inquiry command. The client driver supplies a _BRB_INQUIRY structure with the LAP address on which to base the inquiry access code and a multiplier to calculate the timeout for the inquiry request. The port driver immediately returns a success status. When the inquiry has finished processing, the port driver calls the client driver's HCI inquiry callback routine with the result of the inquiry.

BrbInquiryCancel
Cancels a previously issued inquiry command. The client driver supplies a _BRB_INQUIRY_CANCEL structure with the size and request type.

BrbGetDeviceList
Gets the current list of discovered devices up to the maximum number specified by the Bluetooth client driver. The client driver supplies a _BRB_GET_DEVICE_LIST structure with a pointer to a BthDeviceInfoList of adequate size. On success, the port driver writes a list of devices to the BthDeviceInfoList.

BrbGetLocalBdAddress
Gets the Bluetooth device address of the local radio. The client driver supplies a _BRB_GET_LOCAL_BD_ADDRESS structure with a pointer to a BD_ADDR structure. On success, the port driver writes the address of the local radio to the BD_ADDR.

BrbGetSignalParameters
Gets the Receiver Signal Strength Indicator (RSSI) and transmit power level information for the local radio. The client driver supplies a _BRB_GET_SIGNAL_PARAMETERS structure with an initialized header. On success, the port driver writes signal parameters to this structure.

BrbHciUnknownCmd
Passes an unknown HCI command to the miniport. The client driver supplies a _BRB_HCI_UNKNOWN_CMD structure with the opcode and any parameters required by the command.

BrbFlush
Causes the HCI controller to discard all data that is currently pending for transmission over the specified connection. The client driver supplies a _BRB_FLUSH structure with a handle to the connection for which to discard pending data.

BrbScoOpen
Opens a synchronous connection-oriented (SCO) link for isochronous requests such as audio and (eventually) video. The client driver supplies a _BRB_SCO_OPEN structure with a handle to the connection to open.

BrbScoClose
Closes a previously opened SCO link. The client driver supplies a _BRB_SCO_CLOSE structure with a handle to the connection to close.

BrbScoRead
Reads data over an isochronous channel. The client driver supplies a _BRB_SCO_READ structure with a handle to the connection and a pointer to a buffer to receive the data.

BrbScoWrite
Writes data over an isochronous channel. The client driver supplies a _BRB_SCO_WRITE structure with a handle to the connection and a pointer to a buffer that contains the data to write.

BrbL2caConnectReq
Opens a Logical Link Control and Adaptation Protocol (L2CAP) connection to the specified target device. The client driver supplies a _BRB_L2CAP_CONNECT_REQ structure with Protocol Service Multiplexer (PSM) and Bluetooth address of the target device. On success, the port driver returns a handle to the connection.

BrbL2caConnectRsp
An L2CAP server sends this request in response to an L2CA_ConnectInd event. The server supplies a _BRB_L2CAP_CONNECT _RSP structure with a handle to the connection.

BrbL2capDisconnectReq
Disconnects an established L2CAP connection. The client driver supplies a _BRB_L2CAP_DISCONNECT_REQ structure with a handle to the connection.

BrbL2capDisconnectRsp
An L2CAP server sends this request in response to a request to disconnect an L2CAP connection. The server supplies a _BRB_L2CAP_DISCONNECT_RSP structure with a handle to the connection.

BrbL2capAclTransfer
Transfers Asynchronous Connection-Less (ACL) link data over an L2CAP connection. The client driver supplies a _BRB_L2CAP_ACL_TRANSFER structure with the local channel ID of the connection and a pointer to a buffer for the data. On success, the port driver reads outgoing data from the buffer or writes incoming data to the buffer, depending on the direction of the transfer.

BrbL2capConfigReq
REWRITE: Configures an L2CAP connection for quality of service parameters. The client driver supplies a _BRB_L2CAP_CONFIG_REQ structure with a handle to the connection and the parameters to configure.

BrbL2capConfigRsp
An L2CAP server sends this request in response to a request to configure an L2CAP connection for quality of service. The server supplies a _BRB_L2CAP_CONFIG_RSP structure with a handle to connection, the result of the operation and the parameters that were configured.

BrbSearchServiceRecord
Issues a Service Discovery Protocol (SDP) service record search request. The client driver supplies a _BRB_SEARCH_SERVICE_RECORD structure with a list of search patterns and the maximum number of service record handles to return. On success, the port driver writes a list of record handles that match the search patterns.

BrbSearchServiceAttribute
Issues an SDP service attribute search request. The client driver supplies a _BRB_SEARCH_SERVICE_ATTRIBUTE structure with a list of attribute IDs to search for and a pointer to a buffer in which to return the attributes found. On success, the port driver writes a list of attributes to the buffer

BrbSearchServiceRecordAttribute
Issues a request to search for both service record and service attribute. The client driver supplies a _BRB_SEARCH_SERVICE_RECORD_ATTRIBUTE structure with a list of search patterns and attribute IDs to search for. On success, the port driver writes a list of record handles that match the search patterns and a list of attributes. This request is the equivalent of issuing BrbSearchServiceRecord and BrbSearchServiceAttribute requests to the same target device.

Comments

A Bluetooth client driver uses BRBs to communicate with the Bluetooth port driver.

To send a BRB, the driver creates an IRP_MJ_INTERNAL_DEVICE_CONTROL request with the I/O control code IOCTL_INTERNAL_BTH_SUBMIT_BRB and a pointer to an initialized BRB at **Parameters.Others.Argument1** in the I/O stack location of the IRP.

Before a client driver can send a BRB, it must register certain required callback routines and can register additional optional callback routines. To register callback routines, the client driver issues a query-interface IRP (IRP_MJ_PNP with IRP_MN_QUERY_INTERFACE) with a pointer to an initialized BTHPORT_INTERFACE structure that contains pointers to the client driver's callback routines. The port driver calls the appropriate routine after it finishes processing an asynchronous BRB such as an HCI inquiry request.

_BRB_HEADER

```
struct _BRB_HEADER {
  ULONG Length;
  BRB_TYPE Type;
  ULONG Flags;
  NTSTATUS Status;
  PVOID Context[BTHPORT_CONTEXT_SIZE];
};
```

Describes the length in bytes, request type, and completion status for a Bluetooth request.

Members

Length
The size of the BRB, in bytes.

Type
The BRB request type. The constant value specified for **Type** indicates which structure within the union is valid, as indicated in the following table:

Type	Union substructure
BRB_HCI_INQUIRY	_BRB_INQUIRY
BRB_HCI_CANCEL_INQUIRY	_BRB_INQUIRY_CANCEL
BRB_HCI_GET_DEVICE_LIST	_BRB_GET_DEVICE_LIST
BRB_HCI_GET_LOCAL_BD_ADDR	_BRB_GET_LOCAL_BD_ADDRESS
BRB_HCI_GET_SIGNAL_PARAMETERS	_BRB_GET_SIGNAL_PARAMETERS
BRB_HCI_FLUSH	_BRB_FLUSH
BRB_HCI_UNKNOWN_CMD	_BRB_HCI_UNKNOWN_CMD
BRB_SCO_OPEN	_BRB_SCO_OPEN
BRB_SCO_CLOSE	_BRB_SCO_CLOSE
BRB_SCO_READ	_BRB_SCO_READ
BRB_SCO_WRITE	_BRB_SCO_WRITE
BRB_L2CA_CONNECT_REQ	_BRB_L2CA_CONNECT_REQ
BRB_L2CA_CONNECT_RSP	_BRB_L2CA_CONNECT_RSP
BRB_L2CA_DISCONNECT_REQ	_BRB_L2CA_DISCONNECT_REQ
BRB_L2CA_DISCONNECT_RSP	_BRB_L2CA_DISCONNECT_RSP
BRB_L2CA_CONFIG_REQ	_BRB_L2CA_CONFIG_REQ
BRB_L2CA_CONFIG_RSP	_BRB_L2CA_CONFIG_RSP
BRB_L2CA_ACL_TRANSFER	_BRB_L2CA_ACL_TRANSFER
BRB_SDP_SEARCH_SERVICE_RECORD	_BRB_SEARCH_SERVICE_RECORD
BRB_SDP_SEARCH_SERVICE_ATTRIBUTE	_BRB_SEARCH_SERVICE_ATTRIBUTE
BRB_SDP_SEARCH_SERVICE_RECORD_ATTRIBUTE	_BRB_SEARCH_SERVICE_RECORD_ATTRIBUTE

Flags
TBD

Status
The completion status of the BRB. The Bluetooth port driver fills in this field on completion of the request.

Context
Reserved for internal use.

Comments

Each Bluetooth command structure contains a _BRB_HEADER structure. A Bluetooth client driver fills in the **Length** and **Type** fields before submitting the BRB to the Bluetooth port driver. On completion, the Bluetooth port driver fills in the **Status** field with the status of the completed request.

A BRB consisting of a standalone _BRB_HEADER structure is reserved for internal use by the Bluetooth port driver.

_BRB_INQUIRY

```
struct _BRB_INQUIRY {
  struct _BRB_HEADER Hdr;
  UCHAR LAP[3];
  UCHAR InquiryTimeoutMultiplier;
  BthDeviceInfoList* pDeviceList;
};
```

Triggers the radio to issue an HCI Inquiry command.

Members

Hdr
Specifies the length in bytes of this BRB and a request type of _BRB_HCI_INQUIRY. On success, the Bluetooth port driver fills in the completion status of the request.

LAP
Specifies the LAP address from which the inquiry access code is derived.

InquiryTimeoutMultiplier
Specifies the multiplier factor for calculating the timeout for the inquiry request. Must be a value from 1 to 48. The value of **InquiryTimeoutMultiplier** is multiplied by 1.28 seconds to calculate the timeout. A timeout value can be from 1.28 to 61.44 seconds.

pDeviceList
Points to a list of BthDeviceInfo structures that contain information for each device. The **pDeviceList->NumOfDevices** member indicates the number of devices in the list.

Comments

This request is asynchronous and returns immediately. When the port driver completes the request, it calls the client driver's HCI Inquiry Event routine, which the client driver previously registered with the port driver through a PnP query-interface request. There can be only one inquiry request at a time.

_BRB_INQUIRY_CANCEL

```
struct _BRB_CANCEL_INQUIRY {
  struct _BRB_HEADER Hdr;
};
```

Cancels a previously issued inquiry command.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_HCI_CANCEL_INQUIRY. On output, the Bluetooth port driver fills in the completion status of the request.

Comments

This request is asynchronous and returns immediately. The HCI Command Complete event is not propagated back to the client driver.

_BRB_GET_DEVICE_LIST

```
struct  _BRB_GET_DEVICE_LIST{
  struct _BRB_HEADER Hdr;
  BthDeviceInfoList* pDeviceList;
};
```

Gets the current list of discovered devices up to the maximum number specified by the Bluetooth client driver in **pDeviceList->NumOfDevices**.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_HCI_GET_DEVICE_LIST. On output, the Bluetooth port driver fills in the completion status of the request.

pDeviceList
Points to a list of BthDeviceInfo structures that contain information for each device. The **pDeviceList->NumOfDevices** member indicates the number of devices in the list.

Comments

Gets the current list of discovered devices up to the maximum number specified by the Bluetooth client driver in **pDeviceList->NumOfDevices**.

_BRB_GET_LOCAL_BD_ADDR

```
struct _BRB_GET_LOCAL_BD_ADDR{
  struct _BRB_HEADER Hdr;
  BD_ADDR* pbtAddress;
};
```

Gets the Bluetooth device address of the local radio.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_HCI_GET_LOCAL_BD_ADDRESS. On output, the Bluetooth port driver fills in the completion status of the request.

pbtAddress
Points to a BD_ADDR structure allocated by the client driver. On success, the port driver writes the address of the local radio to **pbtAddress**.

_BRB_GET_SIGNAL_PARAMETERS

```
struct _BRB_GET_SIGNAL_PARAMETERS{
  struct _BRB_HEADER Hdr;
  UCHAR RSSI;
  UCHAR MaxPowerLevel;
  UCHAR CurrentPowerLevel;
};
```

Gets the Receiver Signal Strength Indicator (RSSI) and transmit power level information for the local radio.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_GET_SIGNAL_PARAMETERS. On output, the Bluetooth port driver fills in the completion status of the request.

RSSI
The Received Signal Strength Indicator in dB.

MaxPowerLevel
The maximum transmit power level in dBm units.

CurrentPowerLevel
The current transmit power level in dBm units.

Comments

On success, the Bluetooth port driver writes the RSSI, maximum, and current power level as signed values to **RSSI**, **MaxPowerLevel**, and **CurrentPowerLevel**, which are unsigned fields. The client driver is responsible for converting these values to signed values.

_BRB_FLUSH

```
struct _BRB_FLUSH{
  struct _BRB_HEADER Hdr;
  USHORT hConnection;
};
```

Causes the HCI controller to discard all data that is currently pending for transmission over the specified connection.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_HCI_FLUSH. On output, the Bluetooth port driver fills in the completion status of the request.

hConnection
Specifies the handle of the connection for which to discard pending transmission data.

_BRB_HCI_UNKNOWN_CMD

```
struct _BRB_HCI_UNKNOWN_CMD{
  struct _BRB_HEADER Hdr;

  BD_ADDR* pbtAddress;
  USHORT CmdOpcode;
  UCHAR ParametersLen;
  UCHAR Parameters[1];
};
```

Passes an unknown HCI command to the miniport.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_HCI_UNKNOWN_CMD. On output, the Bluetooth port driver fills in the completion status of the request.

pbtAddress;
Points to a BD_ADDRESS of a local or remote radio.

CmdOpcode
Specifies the Opcode Command Field (OCF) for the unknown command.

ParametersLen
The size of the parameters array at **Parameters**.

Parameters
An array of parameters to pass with the unknown command.

Comments

Unless the command is destined for the local radio only, this request can be sent only when no L2CAP connections are associated with the requested Bluetooth address.

All HCI commands generated by this request start with 0x3F, which is the vendor-specific Opcode Group Field (OGF).

_BRB_SCO_OPEN

```
struct _BRB_SCO_OPEN{
  struct _BRB_HEADER Hdr;
  BD_ADDR btAddress;
  ULONG PacketType;
  USHORT hConnection;
};
```

Opens a synchronous connection-oriented (SCO) connection on a physical links for isochronous requests such as audio.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_SCO_OPEN. On output, the Bluetooth port driver fills in the completion status of the request.

btAddress
The Bluetooth address of the target device.

PacketType
One of the following packet types: HV1, HV2, or HV3.

hConnection
On input, specifies an ACL handle. On output, the port driver writes an SCO handle to **hConnection**.

_BRB_SCO_CLOSE

```
struct _BRB_SCO_CLOSE{
  struct _BRB_HEADER Hdr;
  USHORT hConnection;
};
```

Closes a previously opened SCO connection on a physical link.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_SCO_CLOSE. On output, the Bluetooth port driver fills in the completion status of the request.

hConnection
Specifies the SCO handle of the connection to close.

_BRB_SCO_READ

```
struct _BRB_SCO_READ{
  struct _BRB_HEADER Hdr;
  USHORT hConnection;
  ULONG BufferSize;
  PVOID Buffer;
  PMDL BufferMDL;
};
```

Reads data from an open SCO channel into a caller-supplied data buffer.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_SCO_READ. On output, the Bluetooth port driver fills in the completion status of the request.

hConnection
Specifies a handle to the open SCO channel over which to read data.

BufferSize
On input, specifies the length of the buffer at **Buffer** or **BufferMDL**, in bytes. On success, the port driver writes the number of bytes actually written to the client-driver-supplied buffer.

Buffer
Points to a buffer allocated by the client driver to receive the data. If **BufferMDL** is used, **Buffer** must be NULL.

BufferMDL
Points to an MDL buffer allocated by the client driver to receive the data. If **Buffer** is used, **BufferMDL** must be NULL.

_BRB_SCO_WRITE

```
struct _BRB_SCO_WRITE{
  struct _BRB_HEADER Hdr;
  USHORT hConnection;
  ULONG BufferSize;
  PVOID Buffer;
  PMDL BufferMDL;
};
```

Writes data to an open SCO channel from a caller-supplied data buffer.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_SCO_WRITE. On output, the Bluetooth port driver fills in the completion status of the request.

hConnection
Specifies a handle to the open SCO channel over which to write data.

BufferSize
On input, specifies the length of the buffer at **Buffer** or **BufferMDL**, in bytes. On success, the port driver writes the number of bytes actually read from the client-driver-supplied buffer.

Buffer
Points to a buffer allocated by the client driver that contains the data to write. If **BufferMDL** is used, **Buffer** must be NULL.

BufferMDL
Points to an MDL buffer allocated by the client driver that contains the data to write. If **Buffer** is used, **BufferMDL** must be NULL.

Comments

A client driver can submit an MDL buffer or a PVOID buffer, but not both in the same request.

_BRB_L2CAP_CONNECT_REQ

```
struct _BRB_L2CAP_CONNECT_REQ{
  struct _BRB_HEADER Hdr;
  USHORT PSM;
  BD_ADDR btAddress;
  PVOID hConnection;
};
```

Opens an L2CAP connection to the specified device address and channel. If BTHPORT_NOT_BLOCKING is set in the BRB header, the client driver should provide an entry point for a **L2CA_ConnectCfm** callback routine. Otherwise, this request will block.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_L2CAP_CONNECT_REQ. On output, the Bluetooth port driver fills in the completion status of the request.

PSM
Specifies the Protocol Service Multiplexer on the target device.

btAddress
Specifies the Bluetooth address of target device.

hConnection
On output, a handle to the connection. The caller must not use this handle for non-blocking calls until after its **L2CA_ConnectCfm** routine is called with the same handle.

_BRB_L2CAP_CONNECT_RSP

```
struct _BRB_L2CA_CONNECT_RSP{
  struct _BRB_HEADER Hdr;
  PVOID hConnection;
  USHORT Result;
  USHORT Status;
};
```

An L2CAP server sends this request in response to an L2CA_ConnectInd event.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_L2CAP_CONNECT_RSP. On output, the Bluetooth port driver fills in the completion status of the request.

hConnection
The L2cap connection handle provided by the Bluetooth port driver in L2CA_ConnectInd.

Result
One of the following result flags:

CONNECT_RSP_RESULT_SUCCESS
CONNECT_RSP_RESULT_PENDING
CONNECT_RSP_RESULT_PSM_NEG
CONNECT_RSP_RESULT_SECURITY_BLOCK
CONNECT_RSP_RESULT_NO_RESOURCES

Status
One of the following status flags:

CONNECT_RSP_STATUS_NO_INFORMATION
CONNECT_RSP_STATUS_AUTHENTICATION_PENDING
CONNECT_RSP_STATUS_AUTHORIZATION_PENDING

_BRB_L2CAP_DISCONNECT_REQ

```
struct _BRB_L2CAP_DISCONNECT_REQ{
  struct _BRB_HEADER Hdr;
  PVOID hConnection;
};
```

Closes the L2CAP connection specified by the connection handle.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_L2CAP_DISCONNECT_REQ. On output, the Bluetooth port driver fills in the completion status of the request.

hConnection
The L2CAP connection handle provided by the Bluetooth port driver.

_BRB_L2CAP_DISCONNECT_RSP

```
struct _BRB_L2CAP_DISCONNECT_RSP{
  struct _BRB_HEADER Hdr;
  PVOID hConnection;
};
```

Responds to a request to disconnect the L2CAP connection specified by the connection handle.

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_L2CAP_DISCONNECT_RSP. On output, the Bluetooth port driver fills in the completion status of the request.

hConnection
The L2CAP connection handle provided by the Bluetooth port driver.

_BRB_L2CAP_CONFIG_REQ

```
struct _BRB_L2CAP_CONFIG_REQ{
  struct _BRB_HEADER Hdr;
  pVOID hConnection;
  L2CapConfigInfo ConfigRequest;
};
```

Configures an L2CAP connection for quality of service.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_L2CAP_CONFIG_REQ. On output, the Bluetooth port driver fills in the completion status of the request.

hConnection
The L2CAP connection handle provided by the Bluetooth port driver.

ConfigRequest
The channel configuration parameters in an L2CapConfigInfo structure. This structure is defined as follows:

```
typedef struct _L2CapConfigInfo{
  USHORT MTU;
  USHORT FlushTO;
  flowspec FlowSpec;
  ULONG Flags;
}L2CapConfigInfo;
```

MTU

Maximum transmission unit.

FlushTO

Flush timeout.

Flowspec

A flowspec structure that defines the Quality of Service parameters listed in the following table.

Member	Description
ULONG ServiceType	One of the following SERVICETYPE values: • NOTRAFFIC No traffic will be transmitted in the specified direction. • BESTEFFORT Default value; indicates reasonable efforts. • SERVICETYPE_GURANTEED Guarantees ability to transmit data at the token rate.
ULONG TokenRate	The token transmission rate in bytes per second.
ULONG TokenBucketSize	The token bucket size in bytes.
ULONG PeakBandwidth	The peak bandwidth in bytes per second.
ULONG Latency	The transmission latency in microseconds.
ULONG DelayVariation	The delay variation in microseconds.

Flags

Flags that specify whether individual configuration options are required or hints.

Comments

A client driver submits an L2CAP configuration request after the connection request completes successfully. For the default configuration, client drivers should use L2CAP_DEFAULT_CONFIG for MTU/FlushTo fields of the ConfigRequest and SERVICETYPE_BESTEFFORT in the ServiceType of the flowspec structure. Depending on the completion status of this request, the remote device servicing the request might modify the ConfigRequest data.

_BRB_L2CAP_CONFIG_RSP

```
struct _BRB_L2CAP_CONFIG_RSP{
  struct _BRB_HEADER Hdr;
  pVOID hConnection;
  USHORT Result;
  L2CapConfigInfo ConfigRequest;
};
```

Responds to a request to configure an L2CAP connection for quality of service.

Members

Hdr

On input, specifies the length in bytes of this BRB and a request type of _BRB_L2CAP_CONFIG_RSP. On output, the Bluetooth port driver fills in the completion status of the request.

hConnection
The L2CAP connection handle provided by the Bluetooth port driver.

Result
One of the following flags:

CONFIG_STATUS_SUCCESS
CONFIG_STATUS_INVALID_PARAMETER
CONFIG_STATUS_REJECT
CONFIG_STATUS_UNKNOW_OPTION

ConfigRequest
The channel configuration parameters in an L2CapConfigInfo structure. For details, see _BRB_L2CAP_CONFIG_REQ.

_BRB_L2CAP_ACL_TRANSFER

```
struct _BRB_L2CAP_ACL_TRANSFER{
  struct _BRB_HEADER Hdr;
  USHORT LCID;
  USHORT TransferFlags;
  ULONG BufferSize;
  PVOID Buffer;
  PMDL BufferMDL;
};
```

Transfers Asynchronous Connection-Less link (ACL) data over an L2CAP connection.

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_L2CAP_ACL_TRANSFER. On output, the Bluetooth port driver fills in the completion status of the request.

LCID
The local channel ID of the connection.

TransferFlags
One or more of the following flags.

BTHPORT_TRANSFER_DIRECTION(x)
BTHPORT_TRANSFER_DIRECTION_OUT
BTHPORT_TRANSFER_DIRECTION_BIT
BTHPORT_TRANSFER_DIRECTION_IN
BTHPORT_SHORT_TRANSFER_OK_BIT
BTHPORT_SHORT_TRANSFER_OK

The BTHPORT_SHORT_TRANSFER_OK_BIT is set for an L2CAP read request if the buffer received from the remote device is smaller than the buffer allocated by the client.

BufferSize
The length of the buffer at **Buffer** or **BufferMDL**, in bytes. On output, this member will be updated to reflect the total bytes read if the BTHPORT_SHORT_TRANSFER_OK flag was set. Otherwise the port driver will return an error.

Buffer
Points to a buffer allocated by the client driver for the data. If **BufferMDL** is used, **Buffer** must be NULL.

BufferMDL
Points to an MDL buffer allocated by the client driver for the data. If **Buffer** is used, **BufferMDL** must be NULL.

Comments

A client driver sends this request to submit a data buffer to be filled in from or to transmit data to the open channel associated with the LCID.

_BRB_SDP_SEARCH_SERVICE_RECORD

```
struct _BRB_SDP_SEARCH_SERVICE_RECORD{
  struct _BRB_HEADER Hdr;
  USHORT LCID;
  PVOID ServiceSearchPatternBuffer;
  ULONG ServiceSearchPatternSize;
  USHORT MaxServiceRecordCount;
  USHORT TotalServiceRecordCount;
  PULONG ServiceRecordHandleList;
};
```

Issues search request for a Service Discovery Protocol (SDP) service record.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_SDP_SEARCH_SERVICE_RECORD. On output, the Bluetooth port driver fills in the completion status of the request.

LCID
The local channel ID of the connection. If LCID is zero, the search is targeted for the local radio.

ServiceSearchPatternBuffer
Points to a buffer that contains the search pattern list for the request.

ServiceSearchPatternSize
Specifies the size of the search pattern list at **ServiceSearchPatternBuffer**.

MaxServiceRecordCount
Specifies the maximum number of service record handles to return.

TotalServiceRecordCount
On success, indicates the total number of service records that match the search pattern.

ServiceRecordHandleList
On success, points to a list of record handles that match the search pattern.

Comments

This request returns a list of service record handles for records that match a caller-supplied search pattern. The caller is responsible for constructing the search pattern according to the SDP specification.

All SDP requests return data in network byte order (big endian). The caller is responsible for converting the data to little-endian order before processing it.

An SDP search request can be targeted for either local or remote radios. For the local radio, set LCID to zero.

The information contained in the **CoD** member of a BthDeviceInfo structure can be used as a filter to limit the number of devices to connect to for subsequent SDP requests.

BRB_SDP_SEARCH_SERVICE_ATTRIBUTE

```
struct _BRB_SDP_SEARCH_SERVICE_ATTRIBUTE{
struct _BRB_HEADER Hdr;
 USHORT LCID;
 USHORT AttributeIdListSize;
 PVOID AttributeIdList;
 USHORT AttributeListSize;
 PVOID AttributeList;
};
```

Issues an SDP service search request for the specified attributes.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_SDP_SEARCH_SERVICE_ATTRIBUTE. On output, the Bluetooth port driver fills in the completion status of the request.

LCID
The local channel ID of the connection.

AttributeIdListSize
The size of the list at **AttributeIdList,** in bytes.

AttributeIdList
The list of attribute IDs for the search. The caller is responsible for formatting the attribute ID list according to the SDP specification.

AttributeListSize
On input, the size of the buffer allocated by the caller at **AttributeList**. On output, the size of the attribute list in bytes.

AttributeList
On output, the attribute IDs and values found for the list at **AttributeIdList**.

BRB SDP_SEARCH_SERVICE_ATTRIBUTE

```
struct _BRB_SDP_SEARCH_SERVICE_RECORD_ATTRIBUTE{
  struct _BRB_HEADER Hdr;
  USHORT LCID;
  PVOID ServiceSearchPatternBuffer;
  ULONG ServiceSearchPatternSize;
  USHORT AttributeIdListSize;
  PVOID AttributeIdList;
  USHORT AttributeListSize;
  PVOID AttributeList;
};
```

Issues a request to search for both service record and service attribute. This request is the equivalent of issuing a **BrbSearchServiceRecord** and **BrbSearchServiceAttribute**.

Members

Hdr
On input, specifies the length in bytes of this BRB and a request type of _BRB_ SDP_SEARCH_SERVICE_RECORD_ATTRIBUTE. On output, the Bluetooth port driver fills in the completion status of the request.

LCID
The local channel ID of the connection. If LCID is zero, the search is targeted for the local radio.

ServiceSearchPatternBuffer
Points to a buffer that contains the search pattern list for the request.

ServiceSearchPatternSize
Specifies the size of the search pattern list at **ServiceSearchPatternBuffer**.

AttributeIdListSize
The size of the list at **AttributeIdList,** in bytes.

AttributeIdList
The list of attribute IDs for the search. The caller is responsible for formatting the attribute ID list according to the SDP specification.

AttributeListSize
On input, the size of the buffer allocated by the caller at **AttributeList**. On output, the size of the attribute list in bytes.

AttributeList
On output, the attribute IDs and values found for the list at **AttributeIdList**.

BthDeviceInfoList

```
typedef struct _BthDeviceInfoList{
  ULONG NumOfDevices;
  BthDeviceInfo DeviceList[1];
}BthDeviceInfoList;
```

The BthDeviceInfoList structure contains an array of devices.

Members

NumOfDevices
Specifies the number of devices in the array at DeviceList. **NumOfDevices** must be 1 or greater.

DeviceList
The first member of an array of BthDeviceInfo structures. This structure is defined as follows:

```
typedef struct _BthDeviceInfo{
  BD_ADDR btAddress;
  CoD ClassOfDevice;
  UCHAR Name[MAX_NAME_SIZE];
}BthDeviceInfo;
```

BtAddress
The Bluetooth address of the remote device.

ClassOfDevice
The CoD value for the device, as defined in the Bluetooth specification.

Name
The friendly name of the device.

BTHPORT_INTERFACE

```
typedef struct {
  INTERFACE Interface;
  BD_ADDR BtAddress;
  USHORT PSM;
  PFNBTHPPORT_HCI_UnknownEvent Hci_UnknownEvent;
  PFNBTHPPORT_L2CA_ConnectInd L2CA_ConnectInd;
  PFNBTHPPORT_L2CA_ConnectCfm L2CA_ConnectCfm;
  PFNBTHPPORT_L2CA_ConfigInd L2CA_ConfigInd;
  PFNBTHPPORT_L2CA_ConfigCfm L2CA_ConfigCfm;
  PFNBTHPPORT_L2CA_DisconnectInd L2CA_DisconnectInd;
  PFNBTHPPORT_L2CA_DisconnectCfm L2CA_DisconnectCfm;
  PFNBTHPPORT_L2CA_QoSViolation L2CA_QoSViolation;
} BTHPORT_INTERFACE, *PBTHPORT_INTERFACE;
```

A BTHPORT_INTERFACE structure provides the Bluetooth port driver with entry points for a client driver's callback functions.

Members

Interface
An INTERFACE structure that specifies the interface information. This structure is defined in Ntddk.h.

BtAddress
On success, the Bluetooth port driver writes the address of the remote device that caused the Bluetooth protocol service provider to load.

PSM
Specifies the PSM (protocol service multiplexer) of the client driver.

Hci_UnknownEvent
Passes an unknown HCI event to a client for handling. This routine is optional.

L2CA_ConnectInd
Indicates a remote connection request. This routine is required.

L2CA_ConnectCfm
Confirms a connection request. This routine is required.

L2CA_ConfigInd
Indicates a remote configuration request to the local server. This routine is required.

L2CA_ConfigCfm
Confirms a configuration request. This routine is required.

L2CA_DisconnectInd
Indicates a remote disconnect request to the local server. This routine is required.

L2CA_DisconnectCfm
Confirms a disconnect request. This routine is required.

L2CA_QoSViolation
Indicates to the local server that a remote device has violated quality of service. This routine is optional.

Comments

The client driver sends an IRP_MN_QUERY_INTERFACE request with an initialized BTHPORT_INTERFACE structure to register its callback routines with the Bluetooth port driver. **Parameters.QueryInterface.InterfaceType** of the IRP must be GUID_BTHDDI_INTERFACE.

On success, the Bluetooth port driver writes a pointer to a context to **Interface.Context**. The client driver uses the context pointer when incrementing or decrementing the reference count with **Interface.InterfaceReference** and **Interface.InterfaceDereference**.